EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH (CERN)

LHCb-DP-2022-003
December 22, 2022

Graph Clustering: a graph-based clustering algorithm for the electromagnetic calorimeter in LHCb

N. Valls Canudas[1], M. Calvo Gómez[1], X. Vilasís-Cardona[1], E. Golobardes Ribé[1]

[1] *Smart Society Research Group, Engineering Department, La Salle Universitat Ramon Llull, Sant Joan de la Salle 42, 08022 Barcelona, Spain*

Abstract

The recent upgrade of the LHCb experiment pushes data processing rates up to 40 Tbit/s. Out of the whole reconstruction sequence, one of the most time consuming algorithms is the calorimeter reconstruction. It aims at performing a clustering of the readout cells from the detector that belong to the same particle in order to measure its energy and position. This article presents a new algorithm for the calorimeter reconstruction that makes use of graph data structures to optimise the clustering process, that will be denoted Graph Clustering. It outperforms the previously used method by 65.4% in terms of computational time on average, with an equivalent efficiency and resolution. The implementation of the Graph Clustering method is detailed in this article, together with its performance results inside the LHCb framework using simulation data.

Submitted to Eur. Phys. J. C

1 Introduction

LHCb is one of the four main experiments at the LHC at CERN. It consists of a forward-arm spectrometer detector designed to measure the production and decay properties of charm and beauty hadrons with high precision [1, 2]. Starting in 2022, a major upgrade has taken place in order to adapt the luminosity rates of the experiment to the LHC conditions in Run 3. It implies an increment of the instantaneous luminosity by a factor five to $\mathcal{L} = 2 \times 10^{33}\text{cm}^{-2}\text{s}^{-1}$ [3] and a readout rate of 30MHz, or a maximum data rate of 40 Tbit/s for all the subdetectors. In these conditions, collisions which are of interest for many LHCb analyses reach the MHz level in the detector's geometrical acceptance [4]. Therefore, the event selection process is expected to provide an offline-quality reconstruction within the throughput requirements. With the vision of future upgrades implying even tighter time constraints, the LHCb reconstruction needs to be as optimal as possible.

Among the five most time-consuming algorithms in the High Level Trigger 2 (HLT2) sequence of LHCb, the calorimeter reconstruction was the fourth one with the order of 15% of the total cost. To make a significant improvement, the data reconstruction process from this specific sub-detector has been a target to optimise. In this article, a new algorithm for calorimeter reconstruction called Graph Clustering is presented. In terms of execution time, Graph Clustering outperforms the previous method by up to 65.4% with an equivalent efficiency. Overall, it provides an average throughput reduction of 9.8% in the whole HLT2. Furthermore, it is currently the default solution for calorimeter reconstruction in the upcoming Run 3.

This article is aimed to give a background in other reconstruction methodologies used for similar problems, specifically in High Energy Physics, in Section 2. In Section 2.1, an introduction to the Electromagnetic Calorimeter (ECAL) of LHCb is given. Section 3 provides an extensive detail of the Graph Clustering implementation. Finally, in Section 4 a review of the performance of the algorithm is given, followed by a discussion and conclusions.

2 Background

Calorimeter data reconstruction can be understood as a clustering problem, as it aims to group the energy deposits from particles following a set of rules. Classical unsupervised clustering algorithms use extensive recursive functions to create clusters according to metrics related to distance or density [5]. Despite the cluster concept, the calorimeter reconstruction strategy for LHCb has not much in common with classical clustering algorithms, due to the strong physics and execution time requirements.

Focusing on the field of calorimetry in High Energy Physics, the Cellular Automaton has been a benchmark solution for many years [6]. LHCb has been using this strategy for Runs 1 and 2. In 2004 an approach using spanning trees was proposed, using this flexible data structures to exploit the neighbourhood definitions in general calorimeter data [7], but it does not consider the cluster separation needed in LHCb. Graph data structures started to appear in the field with the increasing popularity of deep learning in the form of a neural model based on graphs [8]. Several approaches have used these graph neural networks on layered calorimeters [9, 10] showing promising results on clustering energy deposits in consecutive calorimeter layers. However, the LHCb calorimeter geometry is bi-dimensional. Within this context, other approaches have also used graph neural networks [11] and convolutional neural networks [12, 13] with similar conditions as ECAL in LHCb. That said, the inference of some deep learning models is still not mature enough to be incorporated in the LHCb software framework.

Graph structures have demonstrated to be suited for calorimeter data. Hence, the Graph Clustering algorithm stores the calorimeter digits into graphs and makes use of its flexible neighbourhood properties to define the clusters. Moreover, it follows the same reconstruction principles from the Cellular Automaton strategy, which has proved to give a good performance in terms of reconstruction efficiency.

2.1 Detail of the Electromagnetic Calorimeter

The LHCb experiment has a subset of eight dedicated detectors to acquire data from the particles generated in the LHC collisions. The electromagnetic calorimeter is one of them. Its main purpose is the identification of hadrons, electrons and photons, and the measurement of their energies and positions [14]. The ECAL has a rectangular shape of 7.8×6.3 m^2 and is placed perpendicular to the accelerator beam pipe. The energy measurement area is segmented into individual square-shaped modules. Each module is made from lead absorber plates interspaced with scintillator tiles as active material. The general structure is segmented in three different rectangular shaped regions, as can be seen in Figure 1.

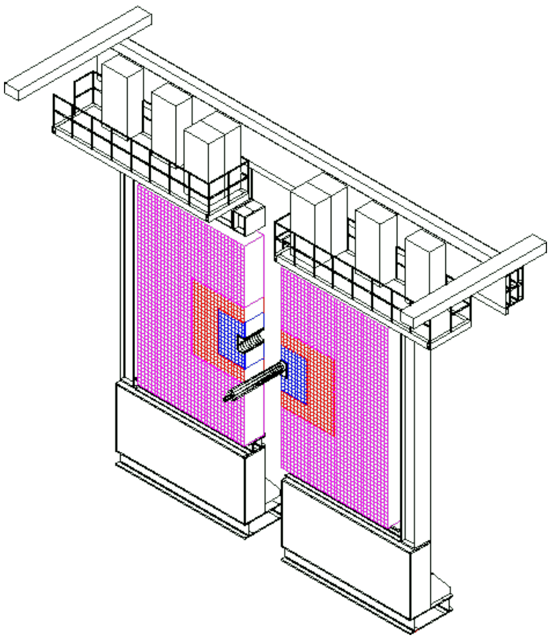

Figure 1: The electromagnetic calorimeter 3d view from behind the detector towards the interaction point [14].

Although all modules have the same size of 12×12 cm^2, the number of readout cells on a module depends on the region. The inner region is the closest to the beam pipe and has the highest occupancy of incident particles. Thus, it has the highest granularity among the three regions, with nine readout cells of 4×4 cm^2 per module. The middle region surrounds the inner one and has four readout cells of 6×6 cm^2 per module. The outer region has a single readout cell of 12×12 cm^2 per module.

The output data obtained from the ECAL modules are the values from each readout cell concerning the accumulated energy deposited by incident particles in a collision event with 12 bit precision. Since particles may deposit energy in more than one readout cell, the energy deposits need to be reconstructed and clustered together with the ones belonging to the same particle. This process is precisely the cluster reconstruction for the calorimeter. The current definition of a calorimeter cluster stands for a 3×3 block of readout cells around an energy peak. Studies have been done regarding the cluster shapes [15] where a combination of 2×2 and swiss-cross cluster shapes show promising performance for high luminosity, although the 3×3 cluster is used as a base for masking other shapes on clusters. Hence, the definition of 3×3 readout cell clusters is maintained through all the regions of the detector.

3 Method

The baseline idea behind the Graph Clustering algorithm is to use graphs as a data structure to store the event digits. It transforms the calorimeter digits into independent graph structures, where only relevant digits for a cluster are contained into isolated graphs. Following graph theory nomenclature, each energy digit from an event is represented as a vertex v in the graph, also called node. The relations between digits, representing links to the same cluster, are defined as directional edges (u, v) between the source digit node u and the target node v. By design, the target nodes of all edges in the graph are the seeds of the reconstructed clusters, where a seed is defined as a local maximum energy digit in the calorimeter grid over a threshold of 50 MeV in transverse energy. With this, the cluster seeds can be easily identified as nodes with only incoming edges. Furthermore, a node can be linked to more than one seed if it is susceptible to have energy deposits from more than one particle. These particular cases are called overlap cells. Overall, the graph derived from an event may contain structures like the example shown in Figure 2.

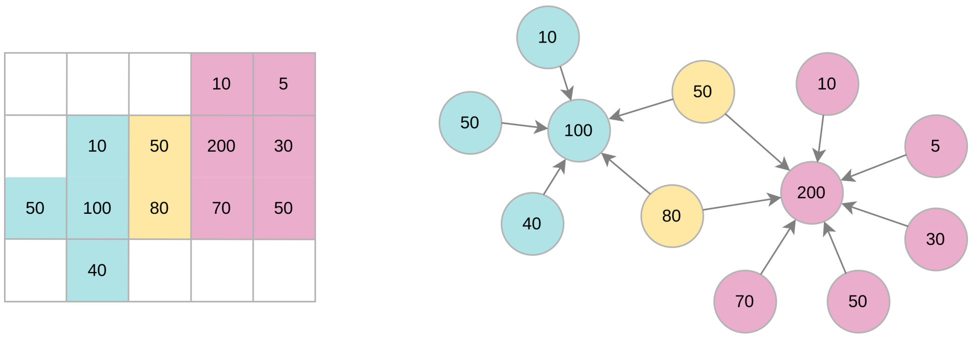

Figure 2: An example of two clusters with overlapping cells on the calorimeter on the left and its graph representation on the right.

The following subsections describe in detail the four steps needed in the Graph Clustering reconstruction process.

3.1 Sorting

To achieve the mentioned representation of the digits, the algorithm needs to make an efficient insertion of the edges into the graph structure. Since all the edges are based on the cluster seeds, the initial key point is to identify seed candidates. As defined previously, a cell in the ECAL grid is considered a seed if it is a local maximum and has a minimum transverse energy value of 50 MeV. A local maximum in this context defines a cell that has the highest energy value among its distance one neighbours in the calorimeter grid. This definition is the same as the one used in the Cellular Automaton algorithm [6].

In order to process the seed candidates of an event in the first place, all the digits above 50 MeV need to be sorted by decreasing traverse energy value. In the proposed algorithm, the sorting is computed using Introspective Sorting [16], which is a hybrid sorting algorithm that combines three different methods to provide fast average performance and optimal worst-case performance.

3.2 Insertion

The role of the insertion step is to build the graph edges between the event digits such that the graph structures of Figure 2 are obtained. A pseudo-code notation of this process is stated in Algorithm 1.

Algorithm 1 Graph insertion

```
1: $G \Leftarrow$ directional weighted graph
2: for each $energy, id \in sortedDigits$ do
3:     if $id$ not inserted in $G$ then
4:         if $id$ is local maxima then
5:             add node $id$ in $G$
6:             for each $n_{energy}, n_{id} \in$ neighbours of $id$ do
7:                 add node $ne_{id}$ and edge $(ne_{id}, id, w = 1)$ in $G$
8:                 if $id$ is a merged $\pi^0$ candidate then
9:                     add $id$ and $n_{id}$ to $mergedPi0$
10:                end if
11:            end for
12:        end if
13:    else if $id \in mergedPi0$ then
14:        $seed =$ first seed from $id$ in $G$
15:        for each $n_{energy}, n_{id} \in$ neighbours of $id$ do
16:            if $energy > n_{energy}$ & $n_{id}$ not in $G$ then
17:                add node $ne_{id}$ and edge $(ne_{id}, id, w = 1)$ in $G$
18:            end if
19:        end for
20:    end if
21: end for
```

It essentially iterates over each sorted digit. That digit may have already been inserted in the graph. If so, this means it is a neighbour of a more energetic digit. In that case, it cannot be a seed since there cannot be two adjacent maxima by construction, except for the case of merged π^0s, which is explained in section 3.2.1. Therefore, that digit is not inserted. On the other hand, if the digit has not yet been inserted on the graph, it can be either a seed or a residual digit, meaning it is not a local maximum and does not have any seed on its neighbourhood. To distinguish between the two, the algorithm checks if that digit is a local maximum. If it is the case, the seed is inserted in the graph together with all its neighbour digits linking them with edges to the seed. The default weight value for all edges is one.

Additionally, if a merged π^0 candidate is identified following the metrics described in section 3.2.1, there is a second seed added to the same cluster. The neighbours of the second seed are also linked with an edge to the first seed if they are not already inserted and if its energy deposit is lower than the energy of the first seed.

At the end of the insertion step, the clusters are already grouped in the graph. However, the overlap cases still need to be processed to adjust the weight of the overlap edges.

3.2.1 Merged π^0 case

One of the reconstruction requirements for the LHCb calorimeter is the correct identification of neutral pions, π^0, which decay into two photons before reaching the calorimeter. Depending on the energy and momentum of the π^0, the two photons arrive at the calorimeter with a certain separation. If the photons are distanced enough to be reconstructed as two separate clusters, it is called a resolved π^0. Otherwise, the two photons may travel very close to each other and reach the calorimeter at one cell distance or less. In that case, the reconstruction is done as a single cluster, since the definition of maxima does not allow two adjacent cluster seeds. When photons are not separable, it is then called merged π^0 case. Hence, the super-cluster from a merged π^0 can be bigger than the 3×3 window around the seed as can be seen in Figure 3.

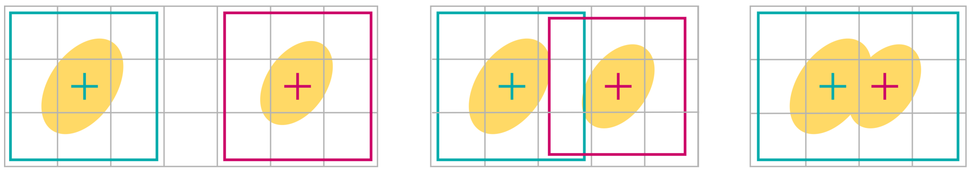

Figure 3: Diagram representation of π^0 cluster cases on the calorimeter. From left to right: the two photons are separable and without overlap, it is a resolved π^0. The two photons are separable but have three overlapping digits, it is however a resolved π^0. The two photons are not separable, it is a merged π^0 and is reconstructed as a single cluster bigger than 3×3.

The cluster shape used in π^0 reconstruction tools in LHCb is a mask of 5×5 cells around the seed [17]. Therefore, the residual energy outside the 3×3 window of a merged π^0 is crucial, as it contains part of the energy from the second photon. That is why the Graph Clustering algorithm adapts the shape of potential merged π^0 candidates, expanding the cluster up to the neighbours of the second most energetic digit in the cluster. The only source of information available to make this selection at run time is the energy deposits of the 3×3 cluster. Therefore, we have studied the relation between the two most energetic digits as a ratio labeled $R1$ to define a threshold on which to make the cluster expansion. Over 46.000 samples of single π^0 deposits from $B^0 \to \pi^+\pi^-\pi^0$ decays simulated using Run 3 conditions have been studied. Figure 4 shows a normalised histogram of the $R1$ ratio for π^0 samples and also for photon samples from $B^0 \to K^*\gamma$ in Run 3 conditions. Given the difference in the two distributions, the threshold for $R1$ is set to 25. The chosen value ensures that the residual energy left outside the cluster is less than 9% for the studied π^0 samples and that the cluster expansion affects an average of 8.2% of the clusters in an event. Further studies have determined that small variations around

10% of the selected threshold value do not significantly change the time complexity of the algorithm nor the π^0 resolution. Moreover, a second threshold for merged π^0 candidates concerning the minimum energy of the cluster seed is set to 1000 MeV, as only high energy π^0s are reconstructed as merged according to the data studied.

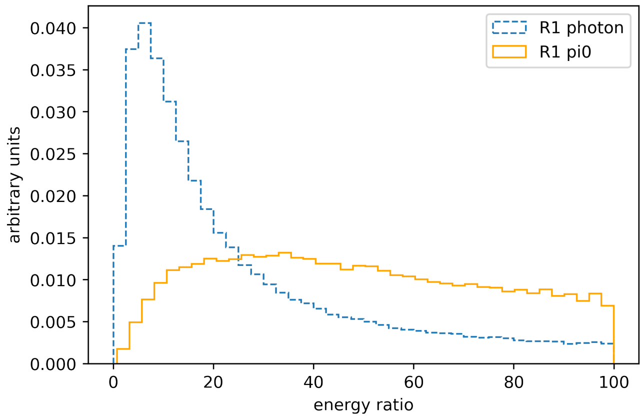

Figure 4: Normalized histogram of the energy ratio between the second most energetic digit and the cluster seed for photon samples and π^0 samples.

3.3 Connected Components

All the clusters are already grouped together after the insertion step. To retrieve them from the graph, we need to search for the sub-graphs where all the nodes are connected to each other by some path, ignoring the direction of edges. This is defined by the weakly connected components of the original graph. In the proposed algorithm, this is implemented as a depth-first search [18], which explores an entire graph exploring all its branches as far as possible before backtracking. Its time complexity is $O(\mid V \mid + \mid E \mid)$ [19] where V is the number of vertices or nodes in the graph and E is the number of edges. Once all the vertices of the graph are visited, the groups of nodes on each weakly connected component are obtained.

3.4 Analysis of Clusters

The processing of each weakly connected component can be done independently of the others, since it contains only the relevant nodes for a cluster. In case there is more than one cluster in a connected component, the overlap fractions are calculated and assigned to the respective edge weights. Either way, the independent clusters are then stored as reconstructed clusters. Algorithm 2 shows a pseudo-code of this step of the reconstruction. Only connected components with more than one vertex are considered as reconstructed clusters. Any isolated vertex is likely to be residual energy deposits from a cluster and should not be considered a reconstructed cluster itself.

In Algorithm 3 a pseudo-code version of the overlap weight calculation is provided. This function iterates through all the vertices in a connected component. It searches for overlap vertices, identified by having two or more output edges, and accumulates the

Algorithm 2 Analysis of connected components

```
1: for each $wcc \in weaklyConnectedComponents$ do
2:     if $wcc.size() > 1$ then
3:         calculate overlap weights (Algorithm 3)
4:         for each $id \in wcc$ do
5:             if $id$ in-edges $> 1$ & $id$ out-edges $== 0$ then
6:                 add $id$ as a cluster seed to $clusters$
7:                 for each $vertex$ connected to $id$ do
8:                     add $vertex$ as entry of $id$ in $clusters$
9:                 end for
10:            end if
11:        end for
12:    end if
13: end for
```

energy of all the connected nodes on all the clusters involved in the overlap, including the energy of the overlapping node equally fractioned for every involved cluster. Then, a weight is computed for every overlapping edge as the fraction between the energy of the target cluster and the sum of all the clusters involved in the overlap.

Algorithm 3 Calculate overlap weights

```
1: $clusterEnergy \Leftarrow$ empty map
2: for each $vertex \in wcc$ do
3:     if $vertex$ out-edges $>= 2$ then
4:         for each $end\_vertex \in vertex$ out-edges do
5:             $energy =$ accumulate energy from the nodes linked to $end\_vertex$.
6:             $energy+ = end\_vertex$ energy / num out-edges.
7:             store $energy$ to $clusterEnergy$
8:         end for
9:         $totalEnergy =$ accumulate $clusterEnergy$ energies with entries $\in vertex$ out-edges
10:        for each $end\_vertex \in vertex$ out edges do
11:            $weight = \frac{clusterEnergy \text{ at } end\_vertex}{totalEnergy}$
12:            set edge $(vertex, end\_vertex, w = weight)$
13:        end for
14:    end if
15: end for
```

At the end of this last step, all the reconstructed clusters from an event are stored.

4 Results

All the algorithm tests have been done within the GAUDI framework [20, 21]. For comparison purposes, this paper evaluates the performance of the Graph Clustering algorithm and the Cellular Automaton algorithm as it has been a benchmark solution until now. Both are tested with the same Monte Carlo data from $B^0 \to K^*\gamma$ simulations using Run 3 conditions.

The quality of the reconstruction in calorimeter algorithms in LHCb is evaluated using metrics of efficiency, energy resolution and position resolution. The efficiency is defined as the fraction between reconstructed particles over reconstructible particles in a set of events. Reconstructible particles are photons that have deposited at least 90% of its energy in the calorimeter cells. On the other hand, reconstructed particles are reconstructible particles matching a cluster from which at least 90% of its energy belongs to that particle. This ratio is later referred to as match fraction. Table 1 shows that Graph Clustering has a higher efficiency than the Cellular Automaton, with 1.02% more reconstructed clusters.

Table 1: Efficiency results in number of reconstructed versus reconstructible clusters from 80,000 $B^0 \to K^*\gamma$ events.

Algorithm	Reconstructible	Reconstructed	Efficiency (%)
Graph Clustering	43234	35313	81.68 ± 0.19
Cellular Automaton	43234	34872	80.66 ± 0.19

On the other hand, the cluster resolution metric aims to measure the difference in energy and position between the reconstructed clusters and the associated Monte Carlo particles. Resolutions are evaluated for γ and π^0 particles. For both cases, we evaluate the difference in position on the X and Y axis and the difference in energy as a percentage. For γ resolution, a total of 80.000 simulation samples of $B^0 \to K^*\gamma$ decays have been used, and another 80.000 samples of $B^0 \to \pi^+\pi^-\pi^0$ decays have been used for π^0 resolution. The study accounts for all the clusters with a match fraction higher than 0.9 since it is the standard match threshold for a cluster to be considered reconstructed in terms of efficiency.

Figure 5 shows the energy distribution for both methods, before any corrections are applied [22], where ΔE stands for the difference in reconstructed energy and truth energy of a cluster. It can be seen that for both γ and π^0 samples the two distributions look very alike. For energy resolution, Graph Clustering is slightly more shifted to negative values, but overall it can be said that the resolution in energy is equivalent to the Cellular Automaton one.

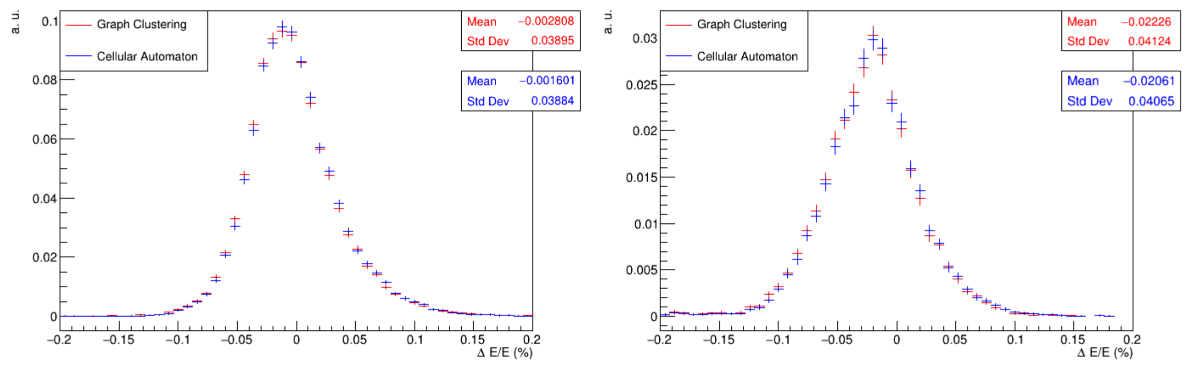

Figure 5: Normalized histograms of the energy resolution with no corrections for clusters with a match fraction over 0.9 using γ samples in the top plot and π^0 samples in the bottom plot.

Regarding the position resolution, Figure 6 shows that the x and y distributions have again an equivalent behavior for both methods. For simplicity, only the π^0 resolutions are shown for position, since the differences with γ samples are minimal.

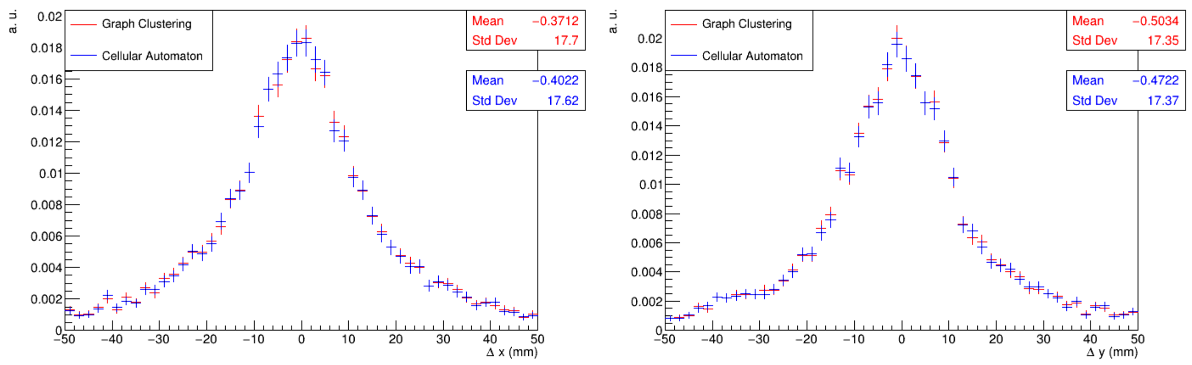

Figure 6: Normalized histograms of the X axis resolution at the top and the Y axis resolution at the bottom. Both using π^0 samples and clusters with a match fraction over 0.9 with no corrections.

Regarding the execution time, it is defined as the time elapsed between the first and the last lines executed in an algorithm. Figure 7 shows a plot of the execution time in arbitrary units as a function of the number of digits per event. The plotted time measurements are obtained as the average measured time from all the events with the same number of digits, from a total of 100.000 events from $B^0 \to K^*\gamma$ simulation. As can be seen from the figure, for events with less than 150 digits, the Cellular Automaton is faster. However, from that point on, Graph Clustering outstands the benchmark algorithm showing a flatter complexity curve. Furthermore, the average number of digits per event from the analysed samples is 1520 digits. At that complexity level, Graph Clustering is 65.4% faster than Cellular Automaton on average.

5 Discussion and Conclusions

Graph Clustering has shown to improve the computational complexity of the calorimeter reconstruction in LHCb. Furthermore, it is the default reconstruction solution for the ongoing Run 3 data taking period. The baseline of the algorithm is to reproduce the same reconstruction steps as in the previously used algorithm, the Cellular Automaton, but with an optimized codification using graph data structures. Hence, it is expected and observed to have similar results compared to the benchmark in terms of efficiency and resolution.

Graphs have demonstrated to be suited for calorimeter reconstruction. Within the proposed implementation, such data structures also provide a flexible interpretation of the neighbour cells in the calorimeter grid. This could also be used to adapt the shape of the clusters to an optimized pattern depending on the region at reconstruction time and significantly accelerate its execution. Currently, the definition of an optimal cluster shape for ECAL clusters is being studied considering pileup and overlap effects as well as precision.

Within the steps of the presented Graph Clustering, as mentioned in section 3.4, the analysis of each connected component is completely independent of the rest of the

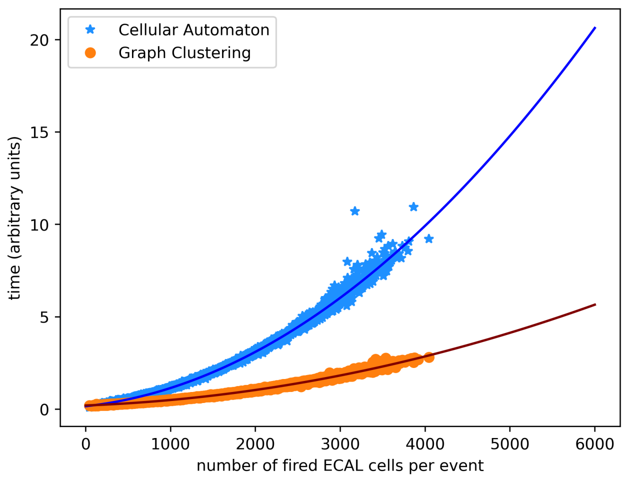

Figure 7: Execution time measured in arbitrary units as a function of the number of digits per event for the Cellular Automaton algorithm and the Graph Clustering algorithm. On top of them, a fitted curve for every algorithm is shown.

graph. Although it is not the most time consuming part of the algorithm, it represents a 27.3% of the total algorithm's execution time, which could benefit from parallel execution. In the context of the first level of the trigger system (HLT1) ran in GPUs, calorimeter reconstruction is at a preliminary stage. The current implementation builds simplified clusters with lower efficiency and resolution than the benchmark. In that direction, there is currently work in progress on adapting the presented Graph Clustering logic to a CUDA algorithm optimized for parallel architectures.

As a final conclusion, the complexity curve that Graph Clustering exhibits makes it a useful alternative for other calorimeters with higher occupancy. Furthermore, the vision of future upgrades in the LHCb calorimeter is a challenging opportunity to test the limits of this algorithm.

Acknowledgements

The authors would like to thank the LHCb computing and simulation teams for their support and for producing the simulated LHCb samples used in the paper. Specially the RTA team for their help with code optimization and the integration into the LHCb framework. This research was funded by Ministerio de Ciencia e Innovación grant number PID2019-106448GB-C32.

References

[1] Alves, A.A., Jr.; Andrade Filho, L.; Barbosa, A.; Bediaga, I.; Cernicchiaro, G.; Guerrer, G.; Lima, H., Jr.; Machado, A.; Magnin, J.; Marujo, F.; et al. The LHCb detector at the LHC. *J. Instrum.* **2008**, *3*, S08005.

[2] LHCb Collaboration. LHCb detector performance. *International Journal of Modern Physics A* **2015** *30(07)*, 1530022.

[3] Bediaga, I.; Chanal, H.; Hopchev, P.; Cadeddu, S.; Stoica, S.; Calvo Gomez, M.; T'Jampens, S.; Machikhiliyan, I.V.; Guzik, Z.; Alves, A.A., Jr.; et al. *Framework TDR for the LHCb Upgrade: Technical Design Report*; Technical Report; LHCb-TDR-012; Geneva, CERN 2012.

[4] Fitzpatrick, C. and Gligorov, V. V. Anatomy of an upgrade event in the upgrade era, and implications for the LHCb trigger. Technical report; LHCb-PUB-2014-027, CERN-LHCb-PUB-2014-027; Geneva, CERN 2014.

[5] Han, J.; Pei, J.; Tong, H. (2022). Data mining: concepts and techniques. *Morgan kaufmann*, Cambridge, 2022.

[6] Breton, V.; Brun, N.; Perret, P. *A Clustering Algorithm for the LHCb Electromagnetic Calorimeter Using a Cellular Automaton*; Technical Report; CERN-LHCb-2001-123; Geneva, CERN 2001.

[7] Mavromanolakis, G. Calorimeter clustering with minimal spanning trees. *arXiv preprint physics/0409039*, 2004.

[8] Scarselli, F.; Gori, M.; Tsoi, A.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE transactions on neural networks* **2008** *20(1), 61-80.*

[9] Ju, X.; Farrell, S.; Calafiura, P.; Murnane, D.; Gray, L.; Klijnsma, T.; Pedro, K.; Cerati, G.; Kowalkowski, J.; Perdue, G.; et al. Graph neural networks for particle reconstruction in high energy physics detectors. *arXiv preprint arXiv:2003.11603*, 2020.

[10] Qasim, S.R.; Long, K.; Kieseler, J.; Pierini, M.; Nawaz, R. Multi-particle reconstruction in the High Granularity Calorimeter using object condensation and graph neural networks. *EPJ Web of Conferences*, **2021**, *251, 03072.*

[11] Qasim, S.R.; Kieseler, J.; Iiyama, Y.; Pierini, M. Learning representations of irregular particle-detector geometry with distance-weighted graph networks. *The European Physical Journal C*, **2019**, *79(7), 1-11.*

[12] Mazurek, M. Deep learning solutions for 2D calorimetric cluster reconstruction at LHCb. *4th Inter-experiment Machine Learning Workshop*, Talk; LHCb-TALK-2020-178; 2020.

[13] Valls Canudas, N.; Calvo Gómez, M.; Golobardes Ribé, E.; Vilasis-Cardona, X. Use of Deep Learning to Improve the Computational Complexity of Reconstruction Algorithms in High Energy Physics. *Applied Sciences*, **2021**, *11(23), 11467*

[14] Omelaenko, O.; Dalpiaz, P.; Guzik, Z.; Spiridenkov, E.; Jarron, P.; Semenov, V.; Ocariz, J.; Khan, A.; Perret, P.; Schneider, O.; et al. *LHCb Calorimeters: Technical Design Report*; Technical Report; LHCb-TDR-002; Geneva, CERN 2000.

[15] Abba, A.; Caponio, F.; Cusimano, A.; Geraci, A.; LHCb, C.; others. *LHCb Particle Identification Upgrade: Technical Design Report*; Geneva, CERN 2013.

[16] Musser, D.R. Introspective sorting and selection algorithms. *Software: Practice and Experience*, **1997**, *27(8), 983–993*

[17] Deschamps, O.; Belyaev, I.; Machefert, F.P.; Pakhlova, G.; Schune, M.H. Photon and neutral pion reconstruction. Technical report, CERN-LHCb-2003-091, Geneva, 2020.

[18] Ginsberg, M. Essentials of artificial intelligence. *Newnes*, **2012**.

[19] Cormen, T.; Leiserson, C.; Rivest, R.; Stein, C. Introduction to algorithms. *MIT press*, **2022**.

[20] The LHCb Collaboration. Upgrade Software and Computing. Technical report, ERN-LHCC-2018-007, LHCB-TDR-017, CERN, Geneva, 2018.

[21] Barrand, G.; Belyaev, I.; Binko, P.; Cattaneo, M.; Chytracek, R.; Corti, G.; Frank, M.; Gracia, G.; Harvey, J.; van Herwijnen, E.; Maley, P.; Mato, P.; Probst, S.; Ranjard, F. GAUDI — A software architecture and framework for building HEP data processing applications. *Computer Physics Communications*, **2001**, *140(1), 45–55.*

[22] Vallier, A. Measurement of the CKM angle gamma in the $B^0 \to DK^{*0}$ decays using the Dalitz method in the LHCb experiment at CERN and photon reconstruction optimisation for the LHCb detector upgrade. PhD Thesis, Université Paris Sud-Paris XI, 2015.